AgriFedSecure: IDS Enabled Federated Learning System for Accurate Soil Moisture Prediction

A dissertation Submitted to Visvesvaraya National Institute of Technology, Nagpur in Partial fulfillment of the Requirements For the award of the degree

MASTER OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

K.Raghuvamshi

(Enrollment no. MT23MCS009)

Under the guidance of

Dr.Hemkumar D

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISVESVARAYA NATIONAL INSTITUTE OF TECHNOLOGY

NAGPUR - 440011, MAHARASHTRA (INDIA)

June 2025

©Visvesvaraya National Institute of Technology (VNIT) 2025

Department of Computer Science & Engineering
Visvesvaraya National Institute of Technology Nagpur

CANDIDATE'S DECLARATION

I, **K.Raghuvamshi**, hereby declare that this dissertation titled **"AgriFedSecure: IDS Enabled Federated Learning System for Accurate Soil Moisture Prediction"** is carried out by me in the Department of Computer Science and Engineering of Visvesvaraya National Institute of Technology, Nagpur. The work is original and has not been submitted earlier whole or in part for the award of any degree/diploma at this or any other Institution / University.

K. Raghuvamshi
Name and signature
Date: 24 Jun 25

CERTIFICATE

This to certify that the dissertation titled **"AgriFedSecure: IDS Enabled Federated Learning System for Accurate Soil Moisture Prediction"** is submitted by **K.Raghuvamshi** in partial fulfillment of the requirements for the award of the degree of Master of Technology in Computer Science and Engineering, VNIT Nagpur. The work is comprehensive, complete and fit for final evaluation.

Dr.Hemkumar D
Assistant Professor
Department of Computer Science & Engineering
Visvesvaraya National Institute of Technology
Nagpur - 440011, India

सहायक प्राध्यापक
Assistant Professor
संगणक विज्ञान एवं अभियांत्रिकी विभाग
Dept. of Computer Science & Engg
वि.रा.प्रौ.सं. नागपुर-10 (भारत)
V.N.I.T., NAGPUR-440010. (India)

Dr. M. P. Kurhekar
Head of the Department
Department of Computer Science & Engineering
Visvesvaraya National Institute of Technology
Nagpur - 440011, India
Date:

विभाग प्रमुख
Head
संगणक विज्ञान एवं अभियांत्रिकी विभाग
Dept. of Computer Science & Engg.
वि.रा.प्रौ.सं. नागपुर-10 (भारत)
V.N.I.T., Nagpur-440010. (India)

Department of Computer Science & Engineering
Visvesvaraya National Institute of Technology Nagpur

NON-PLAGIARISM CERTIFICATE

Certified that M. Tech Dissertation titled **"AgriFedSecure: IDS Enabled Federated Learning System for Accurate Soil Moisture Prediction"** submitted by **K.Raghuvamshi**, Enrollment No. MT23MCS009, ID No. 30159 has been checked for plagiarism using TURNITIN software and the overlap is found within prescribed limits. A summary of the report is given below

ORIGINALITY REPORT

16%	11%	10%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

K.Raghuvamshi
Enrollment Number: MT23MCS009

Dr.Hemkumar D
Assistant Professor
Department of Computer Science & Engineering
Visvesvaraya National Institute of Technology
Nagpur - 440011, India

सहायक प्राध्यापक
Assistant Professor
संगणक विज्ञान एवं अभियांत्रिकी विभाग
Dept. of Computer Science & Engg
वि.रा.प्रौ.सं. नागपुर-10 (भारत)
V.N.I.T., NAGPUR-440010. (India)

ACKNOWLEDGEMENTS

At the outset, I would like to express my sincere gratitude to my supervisor, Dr.Hemkumar D, for offering me the opportunity to work on the project titled "AgriFedSecure: IDS Enabled Federated Learning System for Accurate Soil Moisture Prediction." I am truly thankful for his constant guidance, valuable insights, and encouragement throughout the course of this work. His support helped me navigate challenges, explore new ideas, and improve my research in this emerging area of federated learning. I would also like to thank all the teaching and non-teaching staff of the Department of Computer Science and Engineering for their support and for providing essential resources that facilitated the successful completion of this project.

Special thanks to my friends A.Nithin and K.Pradeep for their unwavering dedication, invaluable knowledge, and constant support, which were instrumental in the successful completion of this project.

I would also like to extend my gratitude to Marshal Murmu sir for his guidance and support throughout the course of this project.

Finally, and most significantly, I want to thank my parents for being a constant source of support, love, faith, and inspiration.

K.Raghuvamshi

ABSTRACT

Smart agriculture is an emerging field that utilizes advance technologies to improve the efficiency, accuracy, and sustainability of farming practices. This project introduces AgriFedSecure, a novel framework that brings together Federated Learning (FL) and Long Short-Term Memory (LSTM) networks to predict soil moisture levels, along with an Intrusion Detection System (IDS) based on XGBoost to protect the system from potential cyber threats. The main advantage of using Federated Learning is that it allows multiple devices or clients to train machine learning models on local data without the need to share that data with a central server. This helps in maintaining the privacy and security of sensitive agricultural information while still enabling collaborative learning.

For soil moisture prediction, the LSTM models were trained locally at two client nodes, achieving a high coefficient of determination (R^2) of 0.91 and 0.95, respectively. These results show the effectiveness of the system in making accurate predictions, which are crucial for planning irrigation and managing water usage efficiently. At the same time, the IDS component, which is powered by the XGBoost algorithm, was able to reach a high detection accuracy of 99%, ensuring reliable protection against malicious attacks. To improve the performance of the IDS, several feature selection techniques were used, including Recursive Feature Elimination with Cross-Validation (RFECV), LASSO Regression (L1 Regularization), and Mutual Information analysis.

Overall, this project demonstrates how combining Federated Learning with powerful machine learning models can lead to secure, efficient, and privacy-preserving solutions in smart agriculture. The proposed AgriFedSecure framework not only improves decision-making for irrigation but also strengthens the cybersecurity of agricultural IoT systems.

Contents

ACKNOWLEDGEMENTS iv
ABSTRACT v

1 Introduction **1**
1.1 Background 1
1.2 Objectives of the Study 3
1.3 Methodology 3
1.3.1 Individual soil moisture prediction models 4
1.3.2 Federated learning model 5
1.3.3 Intrusion detection system model 6
1.4 Structure of the Thesis 7

2 Literature Review **9**
2.1 Research on soil moisture prediction model based on deep learning . . 9
2.2 FLAG: Federated Learning for Sustainable Irrigation in Agriculture 5.0 10
2.3 Improving soil moisture prediction with deep learning and machine learning models 11
2.4 PEFL: Deep Privacy-Encoding-Based Federated Learning Framework for Smart Agriculture 12
2.5 Soil Moisture Prediction Using XGBoost 13

3 Preliminaries And Problem Statement **14**
3.1 Federated learning 14
3.2 Long short term memory 16
3.3 Intrusion detection system in federated learning 20
3.4 Problem Statement 22

4 AgriFedSecure Framework **25**
4.1 Local model building 26
4.2 Federated learning model building 28
4.3 IDS model building 32

5 Results and Discussions **37**
5.1 Performance of local models 38
5.2 Performance of AgriFedSecure framework 41
5.3 Performance of intrusion detection system 43

6 Conclusion **46**
6.1 Conclusion and prospects for future Work 46

Bibliography **49**

List of Figures

1.1 Flowchart of the Methodology 3
1.2 Flowchart of the Local model 4
1.3 Flowchart of the Federated learning model 5
1.4 Flowchart of the IDS model 6

3.1 Federated Learning Architecture 15
3.2 Recurrent Neural Network Having Loop 17
3.3 Broader View Of Recurrent Neural Network 18
3.4 Recurrent Neural Network 19
3.5 Long Short Term Memory 19
3.6 AgriFedSecure Architecture 23

5.1 Predicted vs Actual of Soil Moisture 1 38
5.2 Predicted vs Actual of Soil Moisture 2 39
5.3 Predicted vs Actual of Soil Moisture 3 39
5.4 Predicted vs Actual of Soil Moisture 4 39
5.5 Comparison Between Proposed Model vs Existing Model 40
5.6 Client One Results 41
5.7 Client Two Results 42
5.8 Comparison Between Fl Based Approach vs Non Fl Based Approach 43
5.9 Comparison of four different Feature selection methods 43
5.10 Evaluation of Intrusion Detection System using XGBoost 44
5.11 Performance of Model's Prediction Error 44
5.12 Performance of Models Classification 45

List of Tables

5.1 Performance Metrics of Local Models for Soil Moisture Prediction . . 40
5.2 Comparison Between Proposed Model R^2 vs Existing Model R^2 . . . 40
5.3 Federated Learning Results 42

Chapter 1

Introduction

1.1 Background

Smart agriculture has emerged as a transformative approach to address challenges such as rapidly growing population, climate change, and diminishing natural resources,utilizing advanced technologies to revolutionize farming practices. Central to smart agriculture is the Internet of Things (IoT) Choudhary et al. [2025], Hemkumar [2024] which enables data collection and provides actionable insights for precision farming. However, raw data alone cannot drive decision-making effectively. Artificial Intelligence (AI) and Machine Learning (ML) methods process this data to identify patterns, optimize resource allocation, and automate farming processes, significantly reducing manual intervention and improving efficiency Padhiary and Kumar [2025].

Soil moisture is a vital determinant of agricultural productivity, directly affecting water availability for plant and crop yield Tran et al. [2023]. If soil moisture levels fall below the optimal range, plants undergo water stress, which hampers root respiration, limits nutrient absorption, and inhibits growth.On the other hand, excessive soil moisture can lead to overwatering, causing waterlogging and reduced oxygen availability. Particularly in water-scarce regions, precision irrigation systems guided by accurate soil moisture predictions are essential to prevent these extremes, conserve water, and maximize productivity. As a result, monitoring and predicting soil moisture has become a critical focus in modern agriculture to sustain healthy crop growth and ensure resource efficiency Yu et al. [2021].

To achieve accurate soil moisture prediction, traditional ML models of-

ten require centralized data collection, which presents several challenges. First, data privacy Hemkumar et al. [2021, 2020] and security become significant concerns, as sensitive agricultural data shared on centralized servers is vulnerable to breaches, which leads to market manipulation or insurance providers might alter premiums based on predicted farming outcomes Bera et al. [2024]. To end this, Federated Learning (FL) provides an effective resolution to these obstacles by facilitating decentralized model training across various devices or locations without necessitating the transfer of raw data to a central server. Additionally, FL aggregates model parameters from multiple participants to create an enhanced and generalized model, benefiting from the diversity of data across different farms and regions. This aggregation ensures that the resulting model is more robust and capable of accounting for varying environmental and soil conditions, which is crucial for accurate soil moisture prediction.

Furthermore, the IoT devices used in the federated learning framework introduce new challenges related to cybersecurity. IoT networks are vulnerable to cyberattacks, such as unauthorized access, data tampering, and malware intrusions, which allow attackers to manipulate critical farming operations, such as overwatering or underwatering of crops, causing significant damage to yield, resource wastage, and financial losses for farmers. Therefore, an Intrusion Detection System (IDS) is an essential element in this context, monitoring network traffic to identify and prevent potential threats. Also ensures that agricultural systems remain secure, reliable, and resilient, even in the face of evolving cyber threats.

This Study introduces AgriFedSecure, a novel framework integrating Federated Learning (FL) with Long Short-Term Memory (LSTM) networks for precise soil moisture prediction and XGBoost-powered IDS for robust cybersecurity. By tackling the interconnected challenges of safeguarding data and enhancing system protection, this framework highlights the potential of integrating advanced analytics and cybersecurity measures to build sustainable, resilient, and future-ready smart agricultural systems.

1.2 Objectives of the Study

The main objective of this study is to create a Federated Learning framework using LSTM for accurate and privacy-preserving soil moisture prediction while enhancing system effectiveness and security through the integration of IDS.

- Development of individual soil moisture prediction models for each client to estimate soil moisture levels using LSTM (Long Short-Term Memory) networks.
- Development of Federated Learning Framework that consists of two clients and one server by utilizing LSTM networks for data analysis and a FedAvg operation for model aggregation.
- Development of Intrusion Detection System (IDS) to address cybersecurity challenges in smart agriculture like remote-to-local (R2L) attacks, probing attacks,etc.

1.3 Methodology

The overall methodology is divided into three major parts :

- Local models
- Federated learning model
- Ids model

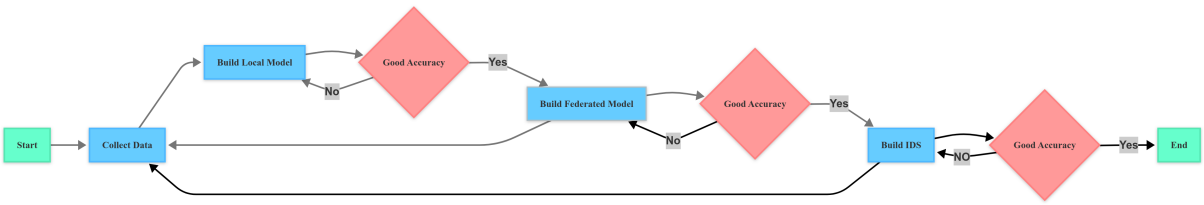

Figure 1.1: Flowchart of the Methodology

1.3.1 Individual soil moisture prediction models

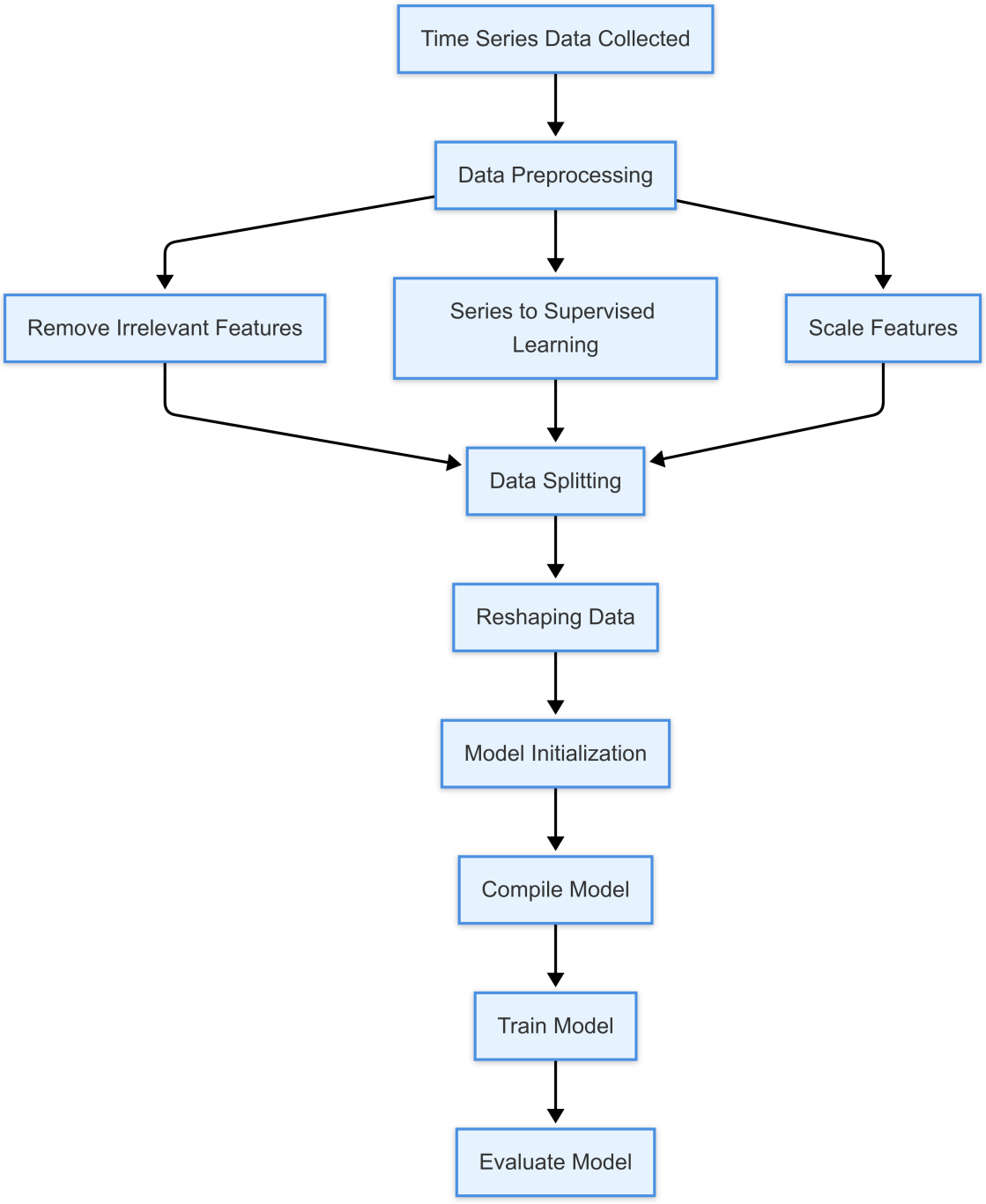

Figure 1.2: Flowchart of the Local model

1.3.2 Federated learning model

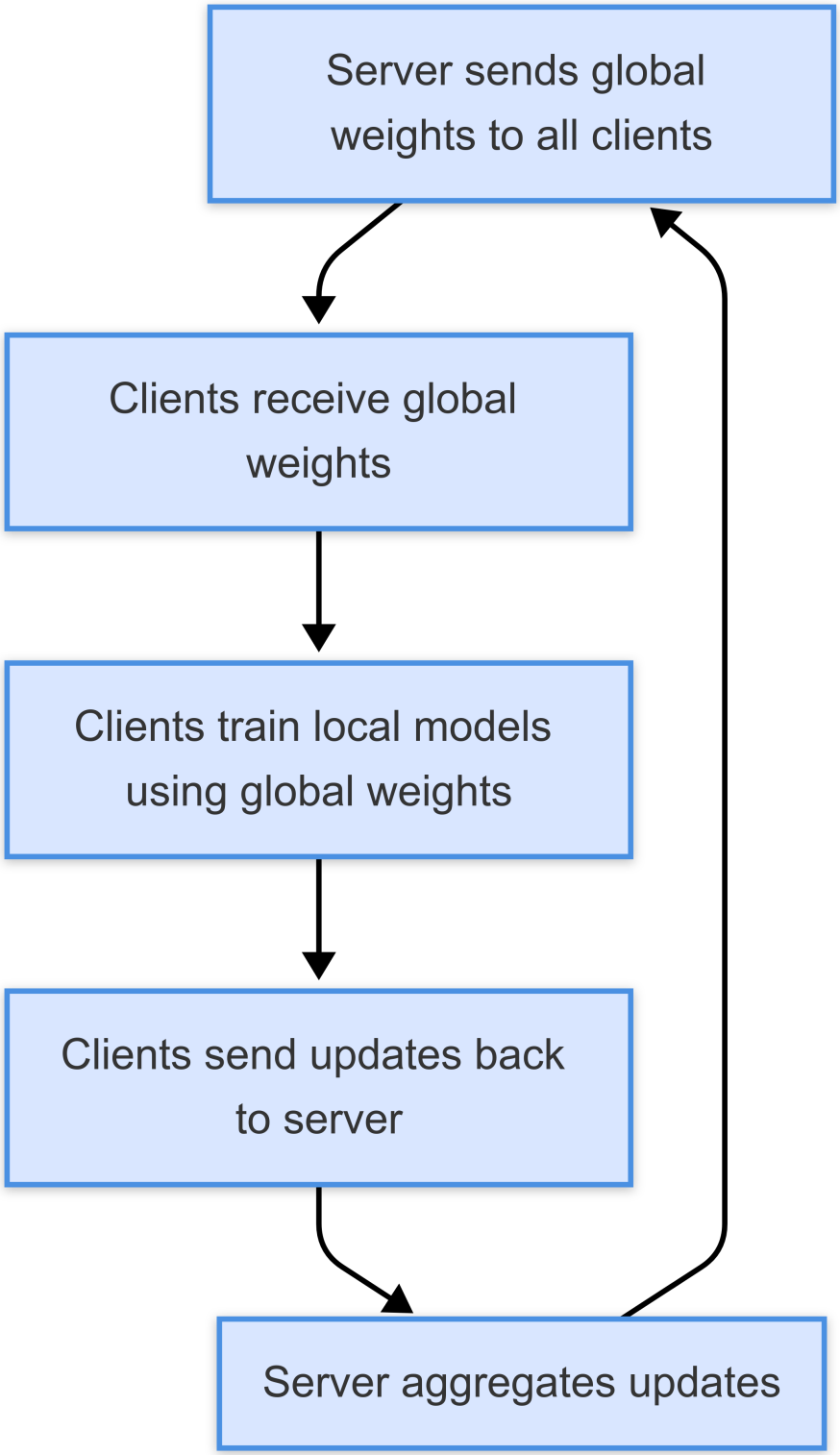
Server sends global weights to all clients
Clients receive global weights
Clients train local models using global weights
Clients send updates back to server
Server aggregates updates

Figure 1.3: Flowchart of the Federated learning model

1.3.3 Intrusion detection system model

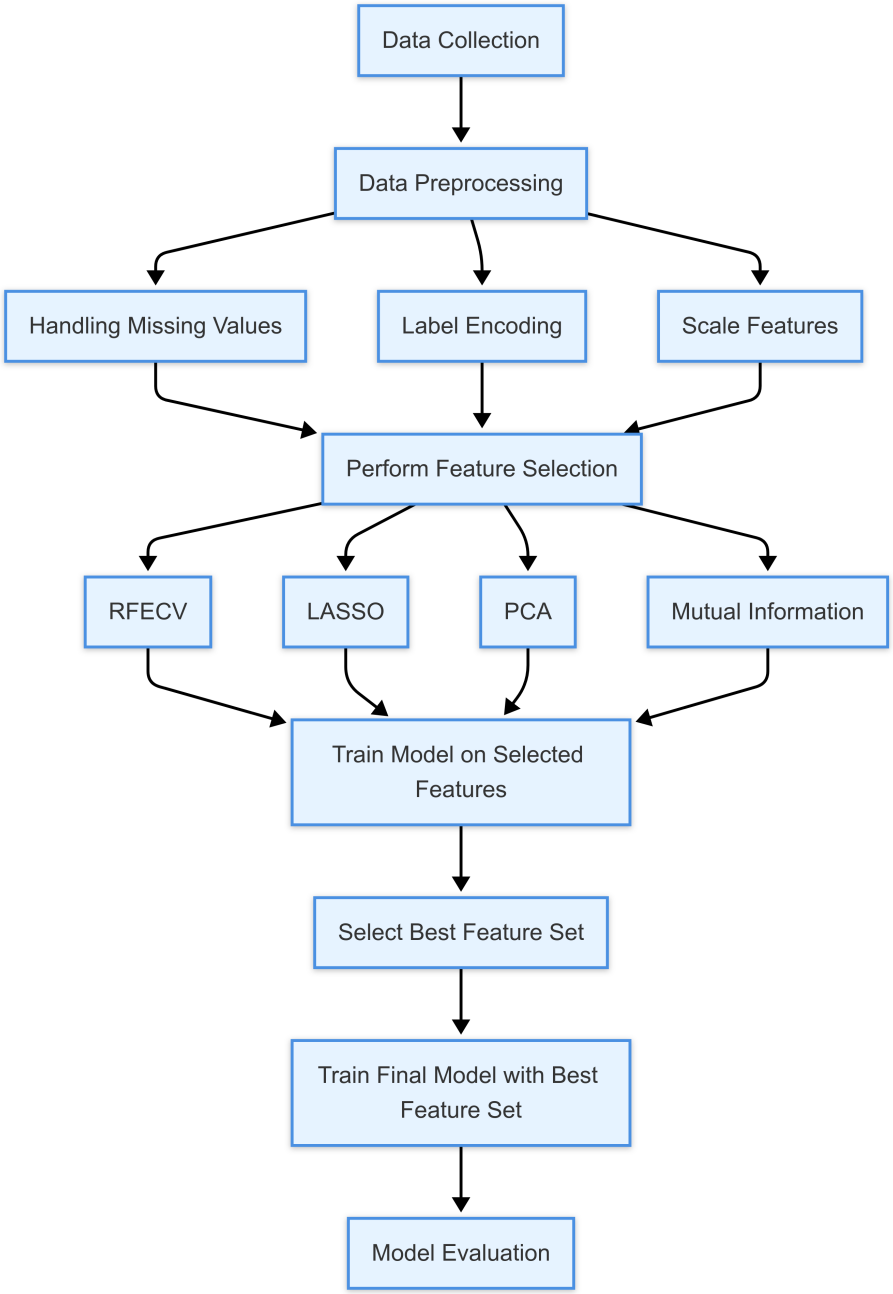

Figure 1.4: Flowchart of the IDS model

The development process began with building a good local model for soil moisture prediction. Once the local model achieved good accuracy, the next step was to implement a Federated Learning model involving two clients and a central server to ensure data privacy while enabling collaborative learning. After successfully building the federated model, an Intrusion Detection System (IDS) was developed to protect the system from potential cyber threats and enhance overall security.

1.4 Structure of the Thesis

Chapter 1: Introduction

This chapter presents the problem statement, the approaches used to address the identified challenges, and overall methodology followed in the project. It outlines how the system was designed and developed, starting from building an accurate local models, extending to a federated learning setup to preserve data privacy, and finally implementing a secure intrusion detection system.

Chapter 2: Literature Review

This chapter presnts the literature review of AgriFedSecure framework which includes an overview of different machine learning and deep learning techniques applied in this field, highlighting their advantages and drawbacks. The review also identifies existing research gaps, which helped in shaping the development of the proposed framework.

Chapter 3: Preliminaries And Problem Statement

This chapter gives an overview of the main technologies used in this framework. It explains why each one is used and what benefits they offer. It gives a brief introduction to LSTM, federated learning, and intrusion detection systems in federated learning.

Chapter 4: AgriFedSecure Framework

This chapter provides a complete overview of the AgriFedSecure Framework and the techniques used for data analysis. It describes how LSTM, Federated Learning, and XGBoost are applied to develop the models. The chapter also walks through each step of the model-building process in detail.

Chapter 5: Results And Discussions

This chapter presents the full results of the AgriFedSecure framework. It begins by showing the performance of the local models, followed by the results of the federated learning model and the IDS model. and also compares the existing local models with the proposed ones.

Chapter 6: Conclusion And Futurework

This chapter gives an overview of the AgriFedSecure framework also discusses limitations and future possibilities. It explains where the research

can be improved and how those improvements can be made.

Chapter 7: References

This chapter includes a full list of references that were used to support and complete this framework.

Chapter 2

Literature Review

2.1 Research on soil moisture prediction model based on deep learning

in this research Cai et al. [2019], the authors focused on building a good soil moisture prediction model using deep learning techniques to support water-saving irrigation and improve drought management in agriculture. The researchers highlight that soil moisture is influenced by many complex factors, including meteorological conditions and soil structure, making it difficult to model accurately using traditional approaches like linear regression or maths formulas. These older methods often suffer from low accuracy and poor generalization across different regions.

To address this, the authors developed a prediction model using a Deep Neural Network Regression (DNNR). They used meteorological and soil data collected from 2012 to 2016 in three regions of Beijing (Yanqing, Shunyi, and Daxing). The input features for the model include average temperature, air pressure, humidity, wind speed, surface temperature, precipitation, and initial soil moisture. Among these, initial soil moisture showed the strongest correlation with future moisture values and played a key role in model accuracy.

The deep learning model is carefully optimized with two hidden layers (a 7-100-50-1 architecture) and trained using the Adagrad optimization algorithm to improve performance. The model was tested first on data from Yanqing and later validated using data from Shunyi and Daxing to assess its generalization capability. The results demonstrated high accuracy, with the R^2 values reaching 0.98 in Yanqing, 0.97 in Shunyi,

and 0.96 in Daxing. Compared to other models like MLP, SVM, and ANN, the DNNR consistently showed better performance with lower error rates and better prediction stability.

Although the model performed well across multiple regions, the study does have limitations. It is mainly tested in a specific geographic region (Beijing) with a relatively controlled dataset.The model may not perform equally well in areas with different climates, soil types, or agricultural practices unless retrained with new data. Additionally, it does not include security or anomaly detection mechanisms, which would be essential for making the system more robust and trustworthy in smart agriculture applications.

2.2 FLAG: Federated Learning for Sustainable Irrigation in Agriculture 5.0

In this research work Bera et al. [2024], the authors introduced a smart irrigation framework named FLAG. The system is designed to help farmers make accurate and timely irrigation decisions while preserving data privacy and minimizing energy consumption and delay. It combines several advanced technologies such as Internet of Things (IoT), dew computing, edge computing, federated learning, and deep learning models like LSTM and DNN. The IoT sensors are used to collect real-time data from the field, such as soil moisture, air temperature, humidity, and wind speed. Instead of sending this data directly to a central server, which can raise privacy and connectivity issues in rural areas, the system processes it locally on edge devices using federated learning. This approach ensures that raw data never leaves the user's device, enhancing privacy, while only encrypted model updates are shared with the cloud. Dew computing is used to temporarily store data when there is no internet connection, allowing the system to function even in low-connectivity areas. The model was tested using the Wazihub dataset and achieved approximately 99% accuracy in predicting whether irrigation is needed or not. Additionally, the system significantly reduced latency and energy usage compared to traditional cloud-based approaches.

Although the FLAG framework offers high accuracy and efficiency, it lacks a built-in mechanism to detect cyber threats or anomalies in the data. This is a critical limitation, especially because the system de-

pends on multiple distributed data sources which could be targets for attacks or tampering. The authors acknowledge this gap and suggest that adding an intrusion detection system (IDS) is a future direction for improvement. Therefore, FLAG is a promising step towards sustainable and intelligent irrigation but integrating security features like IDS is necessary to make it more robust and reliable in real-world farming environments.

2.3 Improving soil moisture prediction with deep learning and machine learning models

In this study Teshome et al. [2024], the authors developed and tested various artificial intelligence models to predict soil moisture at sub-hourly intervals, aiming to improve irrigation management and water conservation in agriculture. The research was carried out at the Tropical Research and Education Center in Florida, using real-time soil and weather data collected from sweet corn and green bean plots over two years (2020–2022).

They used soil sensors and weather stations to gather over 140,000 data points for each crop. The input data included measurements like soil temperature, electrical conductivity, air temperature, solar radiation, and atmospheric pressure. The models tested included deep learning (DL) networks and traditional machine learning (ML) algorithms such as:XGBoost (XGB),LightGBM (LGB),CatBoost (CB),Random Forest (RF),k-Nearest Neighbors (kNN),LSTM (Long Short-Term Memory).

Each model was trained using 80% of the data and tested on the remaining 20%. They evaluated model performance using standard metrics such as R^2 (coefficient of determination), RMSE (root mean square error), and a combined score called Global Performance Indicator (GPI).

All models performed well but surprisingly, LSTM which is commonly used for time-series prediction, performed worse than all other models, with lower accuracy and higher error.

Although the other models gave good results, the deep learning model (LSTM), which is usually known for learning complex patterns in data, did not perform as well in this study. One possible reason is that

LSTM models often need more computing power and longer training time, which may have affected their performance here.

2.4 PEFL: Deep Privacy-Encoding-Based Federated Learning Framework for Smart Agriculture

In this study Kumar et al. [2021], the researchers introduced a framework called PEFL, which stands for Privacy-Encoding-based Federated Learning, designed specifically for smart agriculture systems. The main goal of this framework is to protect private agricultural data collected from IoT devices. In modern farming, various sensors and smart devices are connected through the internet to monitor environmental conditions such as soil moisture, temperature, and humidity. However, this kind of internet-based connectivity also increases the risk of data breaches and cyberattacks.

To deal with these issues, PEFL uses a two-level data privacy mechanism. The first level uses a perturbation method, which changes the original data into a new format to hide sensitive details. The second level uses a deep learning technique called LSTM autoencoder, which further transforms the data into a compressed and protected format. Once this privacy protection is applied, the framework uses federated learning, where the model training happens locally on edge devices. Only the trained model parameters—not the raw data—are shared with a central server, keeping the original data private.

Additionally, the PEFL system includes a model called FedGRU, a version of the GRU (Gated Recurrent Unit) neural network designed to work with federated learning. This model is used to detect any cyberattacks or unusual behavior in the smart agriculture system. The researchers tested their approach using the ToN-IoT dataset, which contains real-world-like IoT data with simulated attacks. The results showed that PEFL was highly accurate, achieving over 99% accuracy in detecting intrusions.

Even though PEFL achieved excellent results in experiments, the study was conducted in a controlled environment.

2.5 Soil Moisture Prediction Using XGBoost

This implementation Waziup [2025] demonstrates a well-structured approach for predicting soil moisture changes in agricultural fields using XGBoost regression. It handles missing data smartly, creates useful time-based features, and provides individual predictions for multiple fields. Its ability to model moisture dynamics using real-world sensor inputs makes it highly suitable for supporting smart irrigation systems.

The prediction accuracy in this demonstration was quite low, as shown by the R^2 values. In contrast, my AgriFedSecure framework achieved better results by using an LSTM model, which improved the accuracy of the predictions.

Unlike all the reviewed works that primarily focus on singular applications, such as irrigation management, IoT cybersecurity,AgriFedSecure is distinct due to its dual-objective framework.It uniquely addresses both productivity and security challenges in smart agriculture by leveraging Federated Learning (FL) for accurate and privacy-preserving soil moisture prediction and implementing a separate Intrusion Detection System (IDS) to secure IoT networks. While the two models are distinct, the dual-objective framework ensures a comprehensive approach to improving resource efficiency and safeguarding system integrity in smart agriculture.

Chapter 3

Preliminaries And Problem Statement

3.1 Federated learning

Federated Learning (FL) Yang et al. [2021] is an innovative decentralized machine learning paradigm that enables multiple entities, such as IoT devices or organizations, to jointly train a shared model while ensuring their data remains on local devices. Unlike traditional centralized systems that require aggregating data at a single server, FL ensures that data remains on individual devices, addressing significant concerns around privacy and security. Furthermore, FL minimizes communication costs by transmitting only model updates rather than large datasets. This makes it highly efficient for resource constrained environments This approach is particularly suitable for sensitive domains like healthcare, agriculture, and finance, where data privacy and legal compliance are paramount.

The step-by-step process of how Federated Learning operates is described below:

- Initialization of Global Model: A global model is initialized and distributed to all the clients involved.
- Local Training: Each client uses its own data to train the global model locally, generating updates such as gradients or weight modifications.
- Sending Updates: The updates computed by the clients are sent back to the central server.

- Model Aggregation: The server combines the updates received from the clients to improve the global model.
- Iteration: Steps 2 and 4 are repeated iteratively until the global model reaches convergence.

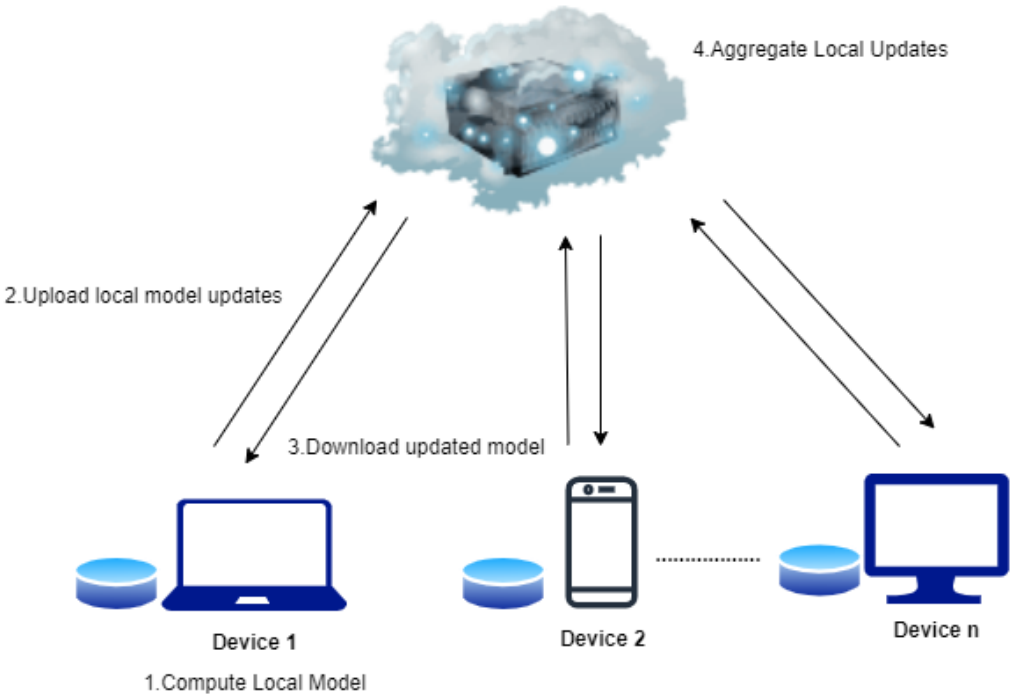

Figure 3.1: Federated Learning Architecture

Federated Learning can be divided into two main types to meet the diverse needs of distributed systems. Horizontal Federated Learning (HFL) applies when clients share an identical feature space but possess distinct data samples, such as farms in various regions collecting similar data attributes. On the other hand, Vertical Federated Learning (VFL) is ideal for situations where clients have the same data samples but differ in feature spaces. The effectiveness of Federated Learning depends on how model updates from clients are aggregated.

Federated Averaging (FedAvg) is a common method that trains a shared model by averaging updates from all participating devices. It aims to minimize overall loss by reducing individual local losses. FedAvg is most effective when all devices have similar data types and computing power.

$$w^{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_k^t$$

- w^{t+1}: updated global model at round $t+1$
- K: number of clients
- n_k: number of data samples on client k
- $n = \sum_{k=1}^{K} n_k$: total number of data samples across all clients
- w_k^t: model trained locally on client k during round t

Fed Prox (Federated Proximal) Allows each device to contribute at its own pace and adds a "proximal" term to prevent large weight changes, helping stabilize training when devices have different data or computing power. Use FedProx when devices have different data distributions or varying hardware capabilities.

$$\min_w h_k(w; w^t) = F_k(w) + \frac{\mu}{2}\|w - w^t\|^2$$

We choose the value of by testing different settings and picking the one that works best.

qFedAvg (Quantile Federated Averaging) Focuses on fair training by penalizing poorly performing devices, ensuring they don't negatively impact the overall model. qFedAvg is most effective when you need fair performance across all devices, especially if some devices have unique data patterns that are hard to learn.

$$\min_w f_q(w) = \sum_{k=1}^{m} \frac{p_k}{q+1} F_k^{q+1}(w)$$

per-FedAvg (Personalized Federated Averaging) Creates a personalized model for each device, tailored to its unique data, useful when devices have different data distributions.Use per-FedAvg when devices have highly diverse data, and a one-size-fits-all model wouldn't work well.

3.2 Long short term memory

When we think or read Olah [2015], we don't start from zero every second. We remember what came before to understand what comes next. Traditional neural networks don't work like this they treat each input as new and separate, without memory.

That's where Recurrent Neural Networks (RNNs) comes in. RNNs are special because they have loops that let them remember past information. This means they can use what they've seen before to help make better decisions now.

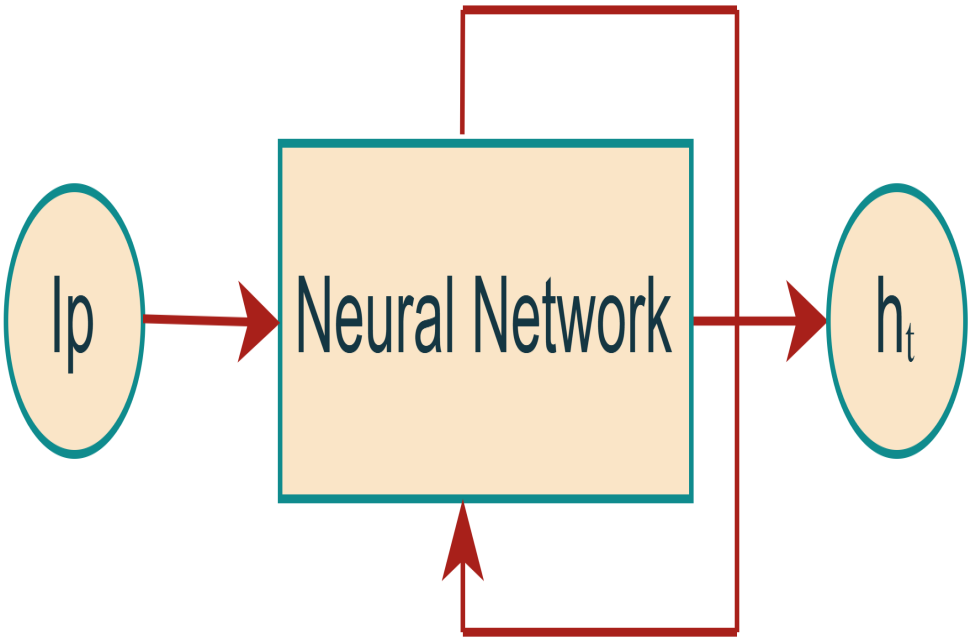

Figure 3.2: Recurrent Neural Network Having Loop

In above Figure 3.2, a neural network unit takes an input Ip and gives an output Op. What makes it special is that it passes some information to the next step using a loop.Although this loop might seem complex, an RNN is really just the same small network repeated over time, with each step passing its memory to the next.

Think of an RNN as a repeating block of the same small network, where each block passes information to the next like links in a chain. This makes RNNs perfect for tasks that involve sequences, like text, speech, or time-series data.

Because of their linked structure as shown in Figure 3.3, recurrent neural networks are well-suited for handling sequences and lists. They are the best type of neural network for time series data, text data, video frames etc..

RNNs have been used successfully in many areas like speech recognition, language translation, and image captioning. A powerful type of RNN called LSTM (Long Short-Term Memory) is especially good at remem-

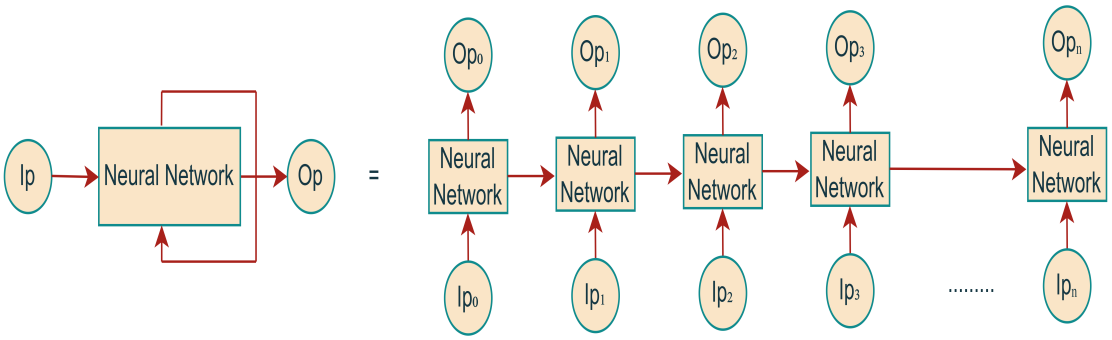

Figure 3.3: Broader View Of Recurrent Neural Network

bering things for a longer time. Most of the best results in RNN-related tasks use LSTMs.

RNNs work well when they only need to remember recent information. But they have trouble with long-term memory they forget things that happened much earlier in the sequence. Because of this, they often make wrong predictions when older information is important. That's why LSTMs were created to help the model remember things for a longer time.

LSTM is a special type of Recurrent Neural Network (RNN) that is good at understanding patterns in time-based data. This makes it well-suited for predicting and Forecasting, where past values influence future readings.

LSTM overcomes vanishing gradient issue of traditional RNNs by using special components called gates. These include the Forget Gate, which decides what information to discard explained in equation 3.1, and the Input Gate, which selects new information to add shown in equation 3.2, equation 3.3. The updated memory of the system is shown in equation 3.4, while the Output Gate controls what the model outputs at each step shown in equation 3.5, equation 3.6. These features help LSTM remember important long-term patterns and work well even with noisy data.

In figure 3.4 and figure 3.5, we can see the difference between rnn and lstms. LSTMs are also built in a chain-like way, but each repeating unit inside them is more complex. Instead of just one layer, each unit has four layers that work together in a unique way.

The cell state, which functions as a belt transferring data throughout

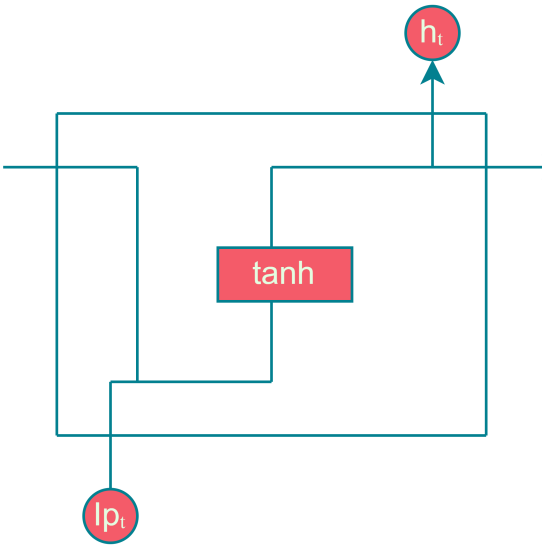

Figure 3.4: Recurrent Neural Network

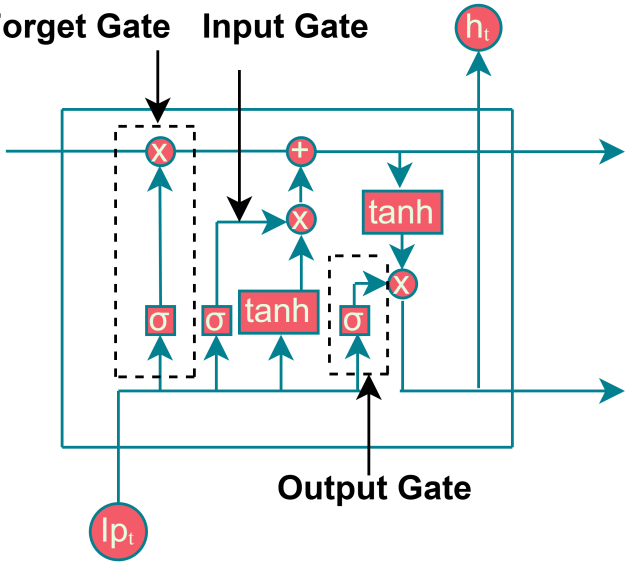

Figure 3.5: Long Short Term Memory

the network, is the most crucial component of an LSTM. It helps convey crucial facts without significantly altering them as it moves through each LSTM unit.

LSTMs can also use specific components called gates to add or remove information from the cell state.

An LSTM's first task is to determine which cell state information to ignore. A forget gate which is a sigmoid layer, is used for this. After examining the current input and the previous output, it assigns values between 0 and 1 to each component of the cell state: 0 indicates the removal of the information, while 1 indicates its retention.

$$f_n = \sigma(w_f \cdot [h_{t-1}; x_t] + b_f) \tag{3.1}$$

Choosing what fresh data to store in the cell state is the next stage in an LSTM. There are two components to this: Which data to update is determined by an input gate, often known as a sigmoid layer. A tanh layer generates potential new values. Ultimately, the cell state is updated by combining these two.

$$i_t = \sigma(w_i \cdot [h_{t-1}; x_t] + b_i) \tag{3.2}$$
$$\tilde{c}_t = \tanh(w_c \cdot [h_{t-1}; x_t] + b_c) \tag{3.3}$$

To obtain the new cell state C_t, it is now necessary to update the previous cell state C_{t-1}. We simply need to put what the previous steps advised us about what to add and what to keep into practice.In order to eliminate unnecessary information, we first multiply the previous state by $f_t \cdot C_{t-1}$. The new data we wish to save, $i_t \cdot \tilde{C}_t$, is then added after being adjusted for importance.

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{3.4}$$

We choose what to output at the end. This is filtered and depending on the cell state. A sigmoid layer first determines which aspects of the cell state to retain. The cell state is then multiplied by the sigmoid output after passing it through a tanh function, which limits values between -1 and 1. Only the portions we desire are included in the final product.

$$o_t = \sigma(w_o \cdot [h_{t-1}; x_t] + b_o) \tag{3.5}$$
$$h_t = o_t \odot \tanh(c_t) \tag{3.6}$$

3.3 Intrusion detection system in federated learning

An Intrusion Detection System (IDS) Al-Shurbaji et al. [2025] is a vital security tool designed to monitor and assess network traffic or system activities to identify and block malicious actions. The working of an Intrusion Detection System involves several key steps:

- Monitors network traffic to identify suspicious activities.
- Analyzes data for abnormal patterns and behaviors.
- Compares activity against predefined rules to detect potential intrusions.

- Sends alerts to administrators when threats are identified.
- Administrators investigate and respond to prevent damage.

The Intrusion Detection System (IDS) can be classified based on their operational environment and detection approach. Host-Based IDS (HIDS) El Bouazzati et al. [2025] focuses on monitoring activities within a single device or host and is particularly effective in detecting insider threats and unauthorized access at the individual system level. In contrast, Network-Based IDS (NIDS) Kumar et al. [2025] monitors traffic across an entire network segment, making it ideal for identifying broader threats such as DoS attacks or unauthorized access attempts across multiple devices. To provide comprehensive protection, Hybrid IDS Lazim and Ali [2025] combines the strengths of both HIDS and NIDS, enabling the monitoring of both host-level activities and network traffic, thereby offering a robust defense mechanism against a wide range of security threats.

Building on the classification of Intrusion Detection Systems (IDS), these systems utilize various detection strategies to identify intrusions effectively. Signature-based detection Ahmed et al. [2025] operates by matching observed activities with a repository of pre-defined attack patterns or signatures. This method is particularly effective at recognizing established threats but might face challenges in identifying novel or emerging attacks. Conversely, anomaly-based detection Arafah et al. [2025] emphasizes detecting deviations from typical behavioral patterns, allowing for the identification of previously unknown or unexpected intrusions. However, this technique may result in higher false positive rates due to variations in normal behavior. By combining the advantages of both approaches, hybrid detection merges signature-based and anomaly-based methods, providing a more reliable and robust solution for intrusion detection. This comprehensive approach ensures that IDS can deliver enhanced protection in diverse and dynamic settings, such as those present in Federated Learning systems.

In the context of Federated Learning (FL), where numerous distributed devices work together to train models, safeguarding the communication and data security becomes crucial. Furthermore, IDS fosters trust among clients by ensuring a secure collaboration environment and preventing financial losses arising from attacks on agricultural systems, such

as soil moisture monitoring, which can lead to resource wastage and increased operational costs. The decentralized nature of FL makes it susceptible to various cyber threats, including data poisoning, model manipulation, and adversarial attacks, as well as various attacks like

Remote-to-Local (R2L) Attacks: Involves unauthorized access originating from a remote device by exploiting weak authentication mechanisms or guessing passwords. This allows attackers to breach sensitive data or take control of IoT devices from a distance Rajkumar et al. [2025].

User-to-Root (U2R) Attacks: Exploits vulnerabilities like buffer overflow to escalate privileges from a user level to root access.Enables attackers to take full control of critical agricultural operations, such as irrigation or drone systems Mounika et al..

Probing Attacks: Includes reconnaissance activities like port scanning to identify open ports or vulnerable devices.Provides attackers with entry points for further exploitation of the system Kumari et al. [2025].

Denial-of-Service (DoS) Attacks: Attackers flood the network with an overwhelming number of requests, such as SYN flood attacks. This disrupts the functionality of essential systems, such as irrigation, monitoring devices, or communication networks Anand et al. [2025].

3.4 Problem Statement

In the AgriFedSecure framework as shown in Figure 3.6, we begin with the assumption that the sensor data transmitted from the farming field remains untampered as it reaches the edge layer. Each device at the edge is equipped with an Intrusion Detection System (IDS), which plays a crucial role in ensuring the integrity of the data. At this stage, each device trains its own model using the locally collected dataset and sends the model updates, specifically gradients or weights, to the central server. Upon receiving the updates, the central server houses an IDS that works to detect any anomalies within the incoming gradients. If no irregularities are detected, the server aggregates the received gradients, which helps enhance the global model. This updated global model is then sent back to the client devices, where the IDS checks again for anomalies. Once confirmed, the clients proceed to train the enhanced model locally on their devices.

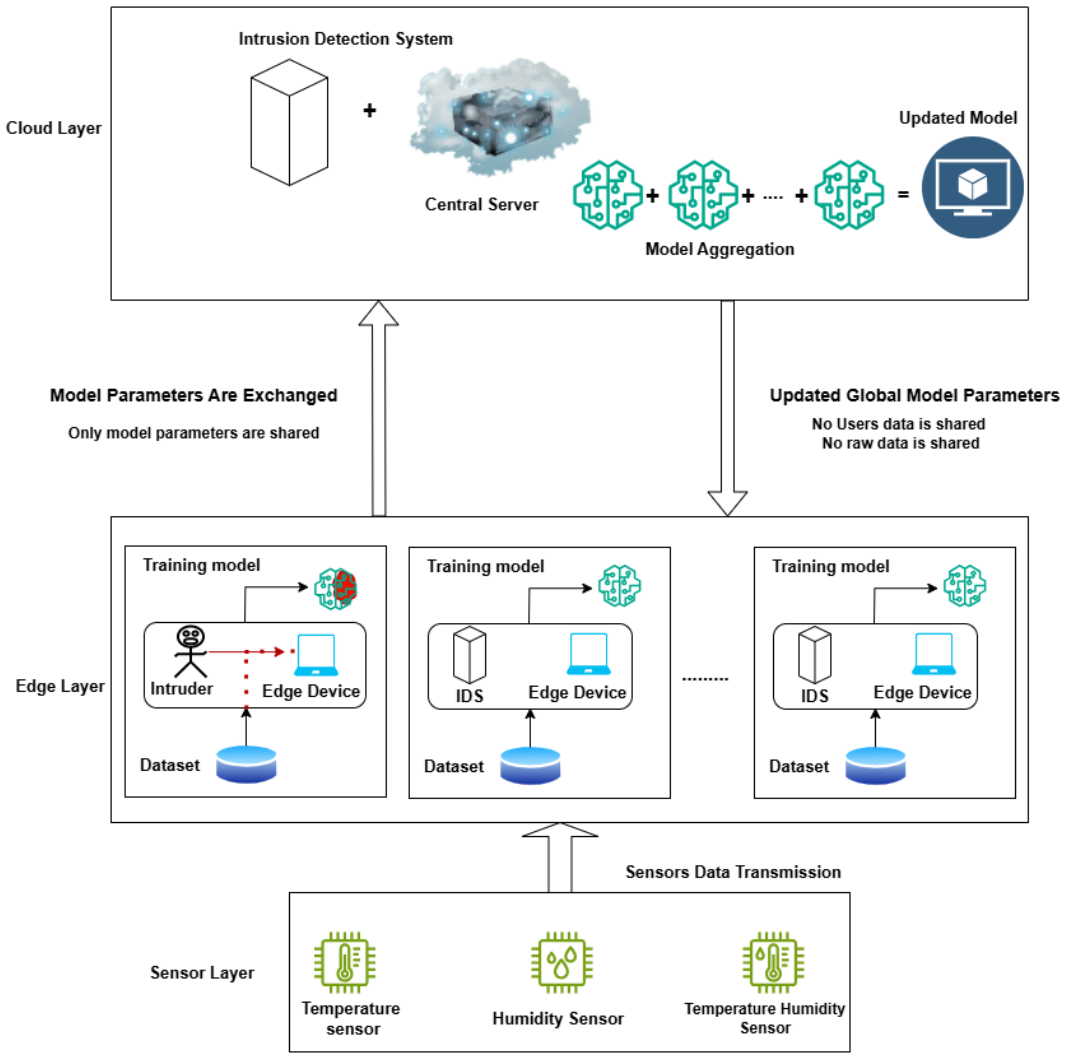

Figure 3.6: AgriFedSecure Architecture

This secure flow ensures that the system maintains data integrity throughout, especially when the IDS is present at both the edge and central server layers. However, a significant problem arises if the IDS is absent at even one of the client devices. Consider the scenario where an attacker tampers with the data at Client 1 as shown in the Figure 3.6. If the gradients from Client 1 are compromised, the faulty gradients will be sent to the central server. The IDS at the central server will only detect these anomalies if it is present. If the central server lacks the IDS, these corrupted gradients will be aggregated with others from the non-compromised devices, resulting in a flawed global model. When this flawed global model is sent back to the clients, the compromised gradients negatively impact the training process on all devices, thus diminishing the overall performance and security of the federated learning

framework.

To mitigate the potential attacks on smart agriculture systems using Federated Learning (FL), a robust Intrusion Detection System (IDS) is designed to detect and respond to various cyberattacks, including DoS, R2L, U2R, and probing. This framework proposes a Federated Learning framework using LSTM for accurate and privacy-preserving soil moisture prediction while enhancing system effectiveness and security through the integration of IDS.

Chapter 4

AgriFedSecure Framework

The framework consists of two primary components: a Federated Learning(FL) framework for predicting soil moisture and an Intrusion Detection System (IDS) to protect smart agriculture systems from cyber threats.

The proposed framework is designed with a three-layer architecture comprising the sensor layer, edge layer, and cloud layer. The development process began with the construction of the edge layer, where we focused on the development of local models where Long Short-Term Memory (LSTM) networks were utilized to predict soil moisture values. These models demonstrated strong predictive performance, achieving high R^2 scores for individual clients. Building on this success, a Federated Learning model was constructed, involving two clients and a central server. The FL framework aggregates local model parameters while preserving data privacy, enabling collaborative learning across distributed clients.

In addition to the FL framework, a robust Intrusion Detection System (IDS) is developed using the XGBoost algorithm to detect and mitigate potential cyberattacks. Advanced feature extraction techniques, including RFECV, Lasso, PCA, and Mutual Information, were utilized to enhance the IDS's accuracy and efficiency. This IDS is essential for ensuring the security of the Federated Learning framework and IoT infrastructure against threats such as DoS, R2L, U2R, and probing attacks.

4.1 Local model building

In this section, we describe the development of the local model, which plays a crucial role in the proposed framework for accurate soil moisture prediction. The local models are built using Long Short-Term Memory (LSTM) networks, a type of recurrent neural network (RNN) specifically designed for sequence prediction tasks. LSTM is well-suited for this application because it effectively handles timeseries data, such as the sequential measurements of soil moisture, temperature, humidity, and other environmental factors that influence soil moisture over time.

LSTM networks are advantageous in scenarios like soil humidity prediction due to their ability to capture long-term dependencies within the data. Unlike traditional feedforward neural networks, LSTM is designed with specialized units, such as input, forget, and output gates, which regulate the flow of information through the network. These gates enable the LSTM to decide what information to remember and what to discard, allowing the network to maintain relevant temporal information over long sequences while preventing the vanishing gradient problem that typically affects traditional RNNs.

In this case, the input gate decides which incoming data such as previous soil moisture readings, temperature, and humidity should be incorporated into the model. The forget gate plays a key role in removing outdated or irrelevant information, ensuring the model maintains an accurate understanding of current soil conditions. By balancing the retention of useful data and the integration of new inputs, the cell state is effectively updated. Finally, the output gate determines what information from the updated cell state should be used for the final prediction. This selective mechanism of retaining and discarding information is essential for accurately forecasting soil humidity, as it allows the model to focus on relevant historical patterns while ignoring noise. By utilizing this architecture, a robust local model is developed that captures the temporal relationships between various environmental factors and soil moisture.

The Algorithm 1 presents the local model using LSTM. This algorithm takes Time-series dataset X, target Y, Number of timesteps T, Learning rate η as inputs and generates Trained local model $\mathcal{M}_{\text{local}}$ as output. The process begins with preprocessing, where irrelevant features like times-

Algorithm 1 Local Model Algorithm

INPUT: Time-series dataset X and target Y; Number of timesteps T; Learning rate η;
OUTPUT: Trained local model $\mathcal{M}_{\text{local}}$; Performance metrics ($R^2$, MSE)

Preprocessing:
  Remove irrelevant features:
    $X' = X - \{\text{timestamp, irrelevant features}\}$.
  Convert data to supervised format: $X_{\text{supervised}}, Y_{\text{supervised}}$.
  Scale features to $[0, 1]$ using MinMaxScaler.
Data Splitting:
  Split data into training and testing sets: $X_{\text{train}}, Y_{\text{train}}, X_{\text{test}}, Y_{\text{test}}$.
Reshape Data:
  Reshape $X_{\text{train}}, X_{\text{test}}$ into 3D format required for LSTM: $(n\_samples, T, n\_features)$.
Model Initialization:
  Initialize LSTM model $\mathcal{M}_{\text{local}}$ with an LSTM layer, Dropout, and Dense output layer.
Compile Model:
  Use loss function $L = \text{MAE}$ and optimizer $\text{Adam}(\eta)$.
Train Model:
  Fit $\mathcal{M}_{\text{local}}$ on $(X_{\text{train}}, Y_{\text{train}})$ for E epochs and batch size B.
Evaluate Model:
  Predict on X_{test}: Y_{pred}.
  Calculate R^2 and MSE for performance evaluation.

tamps and redundant columns are removed, and the data is converted into a supervised learning format using historical time steps as input features. The dataset is then scaled to a range of [0,1] using MinMaxScaler to ensure uniformity and facilitate faster convergence during training (see step no. 1). The preprocessed data is split into training and testing sets and reshaped into a 3D format $(n\_samples, T, n\_features)$, where T represents the number of timesteps, and $n\_features$ corresponds to the features in the dataset (see step no. 2,3).

The LSTM model architecture includes 128 LSTM units, a dropout layer is added to mitigate overfitting, and a dense output layer to predict soil moisture values. The model is compiled using the Mean Absolute Error (MAE) as the loss function, given by equation 4.1, and optimized using the Adam optimizer Rahman et al. [2025] with a learning rate to ensure effective weight updates. After training, the model's performance is assessed using the RÂ² score, calculated as equation 4.2, and the Mean Squared Error (MSE), given by equation 4.3.

$$MAE = \frac{1}{N} \sum_{k=1}^{N} |y_k - \hat{y}_k| \tag{4.1}$$

$$R^2 = 1 - \frac{\sum (y_k - \hat{y}_k)^2}{\sum (y_k - \bar{y})^2} \tag{4.2}$$

$$MSE = \frac{1}{N}\sum_{k=1}^{N}(y_k - \hat{y}_k)^2 \tag{4.3}$$

The Algorithm 1, through the use of LSTM, effectively predicts soil moisture, offering valuable insights for applications like irrigation optimization and sustainable agriculture.

4.2 Federated learning model building

Building upon the promising results achieved by the local LSTM model, the Federated Learning (FL) framework is proposed to further enhance predictive capabilities while addressing the critical need for data privacy in smart agriculture. Unlike traditional centralized approaches, which require unprocessed data to be transmitted to a central server, FL supports decentralized training by enabling individual clients to collaboratively develop a global model while keeping their data stored locally. This method ensures data privacy, reduces communication overhead, and utilizes diverse datasets from multiple clients to construct a more robust and generalized model.

In this framework, two clients, each trained on local soil moisture datasets, and a central server are used to implement the FL process. The FL model aggregates the locally trained model updates from each client using the Federated Averaging (FedAvg) algorithm, which calculates the weighted average of model parameters. This iterative process enables the development of a global model that retains the strengths of local models while maintaining the privacy of individual datasets. By integrating FL into the predictive pipeline, the framework aims to overcome the limitations of centralized systems and offer a scalable, privacy-preserving solution for soil moisture prediction in smart agriculture. Additionally, the privacy-preserving design minimizes data leakage risks, making it highly suitable for sensitive agricultural data.

The algorithm 2 presents the edge layer overview that encompasses the operations performed by each edge device (client). The algorithm 2 takes Global weights W_{r-1}, local data $\mathcal{D}_i$, epochs E, batch size B as inputs and produces Gradients ∇W_i, loss L_i as outputs. The workflow of this layer is systematically described as follows. At the beginning of

Algorithm 2 Client-Side Operations

INPUT: Global weights W_{r-1}, local data $\mathcal{D}_i$, epochs E, batch size B
OUTPUT: Gradients ∇W_i, loss L_i

1. Receive Global Model:
  $W_{r-1} \leftarrow$ Server
2. Preprocess Local Data:
  Load: $\mathcal{D}_i$
  Remove irrelevant features: $\mathcal{D}_i \leftarrow \mathcal{D}_i -$ columns
  Convert time-series data to supervised format:
    $\mathcal{D}_i \leftarrow \text{SupervisedTransform}(\mathcal{D}_i, n_{\text{in}}, n_{\text{out}})$
  Split: $\mathcal{D}_i$ into $\mathcal{D}_{i,\text{train}}$ and $\mathcal{D}_{i,\text{test}}$
  Normalize:
    $\mathcal{D}_{i,\text{train}}, \mathcal{D}_{i,\text{test}} \leftarrow \text{StandardScaler}(\mathcal{D}_i)$
  Reshape into 3D format:
    $\mathcal{D}_{i,\text{train}} \in \mathbb{R}^{N \times T \times F}$
3. Initialize Local Model M_i:
  $M_i \leftarrow$ LSTM(layers, dropout)
  if $r > 1$ Then
    Set model Weights = M_i.setWeights(W_{r-1})
  endif
4. Train Local Model:
  Train M_i on $\mathcal{D}_{i,\text{train}}$ for E epochs, batch size B
5. Compute Gradients:
  $\nabla W_i = \frac{\partial L(M_i, \mathcal{D}_{i,\text{train}})}{\partial W}$
6. Send Results to Server:
  Send ∇W_i and L_i to the server

each training round, the central server sends the global model weights W_{r-1} to the edge devices, where r denotes the current round. Each edge device preprocesses its local dataset D_i by removing irrelevant features to retain only meaningful data columns. The dataset is transformed into a supervised format

$$D_i \rightarrow \text{SupervisedTransform}(D_i, n_{\text{in}}, n_{\text{out}}),$$

Where n_{in} and n_{out} represent the number of input and output timesteps, respectively.

After preprocessing, the dataset is split into training and testing subsets and normalized using standard scaling techniques, ensuring consistency across features.

$$D_{i,\text{train}}, D_{i,\text{test}} \in D_i,$$
$$D_{i,\text{norm}} = \text{StandardScaler}(D_i),$$

The training subset is then reshaped into a three-dimensional format:

$$D_{i,\text{train}} \in \mathbb{R}^{N \times T \times F},$$

Where N is the number of samples, T is the number of timesteps, and F is the number of features. This format is critical for compatibility

with the Long Short-Term Memory (LSTM) architecture used in the framework.

The local model M_i on each edge device is initialized as an LSTM network:

$$M_i = \text{LSTM}(n_{\text{layers}}, \text{Dropout}_{\text{rate}}, \text{Dense}_{\text{units}}),$$

Where n_{layers} specifies the number of LSTM layers, $\text{Dropout}_{\text{rate}}$ mitigates overfitting, and $\text{Dense}_{\text{units}}$ defines the output layer size. If $r > 1$, the global model weights W_{r-1} are incorporated into the local model as M_i.setWeights(W_{r-1}), maintaining the continuity of learning across rounds.

The local model is then trained on $D_{i,\text{train}}$ for E epochs with a batch size of B using the Mean Absolute Error (MAE) loss function and the Adam optimizer η:

$$L = \text{MAE}, \quad \eta = \text{Adam}(\text{learning\_rate}).$$

After training, the gradients ΔW_i are computed as:

$$\Delta W_i = \frac{\partial L(M_i, D_{i,\text{train}})}{\partial W_i},$$

Where $L(M_i, D_{i,\text{train}})$ is the loss function based on model M_i and training data $D_{i,\text{train}}$ and $\frac{\partial}{\partial W}$ is the partial derivative of the loss with respect to weight W_i. Finally, the output of the gradient ΔW_i along with the local loss L_i sent to the central server.

The algorithm 3 presents the cloud layer that takes initial weights W_0, number of clients N, number of rounds, R and learning rate η as the inputs and generates the final global weights W_R as the output. This layer is responsible for managing and coordinating global model updates across participating clients. The process begins with the initialization of the global model weight W_0, which is set to zero. This global weight acts as the starting point for the collaborative learning process.

For each training round r (where $r = 1, \ldots, R$), the cloud server undertakes a series of operations to refine the global model. Initially, the global model weights from the previous round W_{r-1} are distributed to all participating clients $i = 1, \ldots, N$. This ensures that all clients are synchronized and start training their local models using the latest global model.

Algorithm 3 Server-Side Operations

INPUT: Initial weights W_0, number of clients N, number of rounds R, learning rate η
OUTPUT: Final global weights W_R

1. Initialize Global Model:
  $W_0 \leftarrow$ zeros
2. For each round $r = 1, \ldots, R$**:**
  2.1 Distribute Global Model:
    Send W_{r-1} to all clients $i = 1, \ldots, N$
  2.2 Collect Gradients and Losses:
    Receive ∇W_i and L_i from all clients
  2.3 Aggregate Gradients:
    For each layer l:
    $\nabla W_l^{(r)} = \frac{1}{N} \sum_{i=1}^{N} \nabla W_{i,l}$
  2.4 Update Global Weights:
    For each layer l:
    $W_l^{(r)} = W_l^{(r-1)} - \eta \cdot \nabla W_l^{(r)}$
  2.5 Log Progress:
    Store W_r, and compute average client loss:
    $L_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} L_i$
3. Return Final Model:
  Output W_R

Once clients complete their local training, they compute gradients $\nabla W_{i,l}$ and local losses L_i and send them back to the cloud server. The server applies the Federated Averaging (FedAvg) algorithm to compute the global gradient for each model layer l as:

$$\nabla W_l^{(r)} = \frac{1}{N} \sum_{i=1}^{N} \nabla W_{i,l}.$$

Using these gradients, the global weights are updated for each layer l as follows:

$$W_l^{(r)} = W_l^{(r-1)} - \eta \cdot \nabla W_l^{(r)},$$

where η represents the learning rate. This FedAvg-based update ensures that the global model is refined iteratively based on the contributions of all clients.

To monitor the training progress, the cloud server computes the average client loss for the current round:

$$L_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} L_i,$$

and logs the model's progress. These metrics help evaluate the effectiveness of the global model at each training iteration.

Finally, after all training rounds R are completed, the updated global model W_R is returned as the final output. By leveraging the FedAvg algorithm, the cloud layer enables the seamless integration of knowledge from distributed edge devices, ensuring privacy-preserving and efficient training of the global model.

4.3 IDS model building

The Intrusion Detection System (IDS) plays a critical role in safeguarding the Federated Learning (FL) framework against potential cyber threats. In the AgriFedSecure framework, the IDS is incorporated at both the server (cloud layer) and client (edge layer) levels, ensuring a comprehensive security mechanism. This dual-layered approach not only monitors and detects suspicious activities at the local client level but also reinforces security at the central server, enhancing the resilience and reliability of the FL system.

The AgriFedSecure framework integrates Federated Learning (FL) and Intrusion Detection Systems (IDS) to address the dual challenges of predictive modeling and cybersecurity in smart agriculture. The architecture comprises two primary layers: Edge Layer and Cloud Layer, with IDS strategically implemented at both levels to provide end-to-end protection.

Edge Layer:

- The edge layer consists of multiple client devices (e.g., edge devices), each equipped with an IDS module. These devices are responsible for training local models on their soil moisture datasets while ensuring data privacy.
- Integrated IDS Functionality: The IDS continuously monitors network traffic and local activities to identify and prevent attacks such as data poisoning or unauthorized access. By safeguarding the local training environment, the IDS maintains the integrity of model updates sent to the cloud.It protects individual client devices from localized threats such as tampered data injections or unauthorized access.

Cloud Layer:

- The cloud layer acts as the central server, aggregating the model updates from all edge devices to refine the global model.
- Cloud-Level IDS: An IDS at the cloud level monitors communication between the server and edge devices, detecting and mitigating threats such as model tampering, man-in-the-middle (MITM) attacks, or denial-of-service (DoS).It secures the communication pipeline, ensuring the integrity of model updates and protecting against systemic cyber threats.

The IDS system in the edge layer represents the secure data flow from the sensor layer to the end-user in a smart agriculture system, as shown in Figure 3.6. Sensors deployed in the field collect critical environmental data such as soil moisture, temperature, and humidity, which are then transmitted as data packets through the network. The data travels via routers, which direct it efficiently to the desired destination. To ensure security, a firewall acts as the first layer of defense, filtering network traffic and blocking unauthorized access. Beyond the firewall, an intrusion detection system further secures the network by monitoring and analyzing traffic to detect malicious activities or anomalies in real time. Once validated, the data is presented to the user, enabling them to make informed decisions, such as managing irrigation or monitoring soil health. This multi-layered approach ensures data integrity, security, and reliability, which are essential for smart agriculture systems.

The algorithm 4 presents the IDS System that takes input as training data $\mathcal{D}_{\text{train}}$, testing data $\mathcal{D}_{\text{test}}$, threshold τ, boosting rounds B, learning rate η and generates output as an XGBoost model. The process begins with data preprocessing, where the training dataset $D_{\text{train}} = \{X_{\text{train}}, y_{\text{train}}\}$ and testing dataset $D_{\text{test}} = \{X_{\text{test}}, y_{\text{test}}\}$ are prepared. Missing values are handled, categorical features are encoded, and feature standardization is applied to ensure consistency. Standardization is performed using

$$X_{\text{scaled}} = \frac{X - \mu}{\sigma},$$

where μ represents the mean and σ is the standard deviation of each feature.

Feature selection is a critical step to reduce dimensionality and improve model efficiency. Multiple advanced techniques are employed to identify

Algorithm 4 IDS Algorithm

INPUT: Training data $\mathcal{D}_{\text{train}}$, testing data $\mathcal{D}_{\text{test}}$, threshold τ, boosting rounds B, learning rate η
OUTPUT: XGBoost model $\mathcal{M}_{\text{XGB}}$.

Preprocess Data:
    Handle missing values and label encode categorical features
    Standardize features using:
    $X_{\text{scaled}} = \frac{X-\mu}{\sigma}$
Perform Feature Selection:
    RFECV: Identify feature set $\mathcal{F}_{\text{RFECV}}$
    LASSO: Identify feature set $\mathcal{F}_{\text{LASSO}}$
    PCA: Identify principal components $\mathcal{F}_{\text{PCA}}$ capturing 95% variance
    Mutual Information: Identify feature set $\mathcal{F}_{\text{MI}}$
Train Models on Selected Features:
    Train individual models on features from $\mathcal{F}_{\text{RFECV}}$, $\mathcal{F}_{\text{LASSO}}$, $\mathcal{F}_{\text{PCA}}, \mathcal{F}_{\text{MI}}$
    Evaluate models using accuracy A_{test} and ROC-AUC$_{\text{test}}$
Select Best Feature Set:
    Identify $\mathcal{F}_{\text{best}}$ as the feature set with the highest ROC-AUC$_{\text{test}}$
Train Final Model:
    Train XGBoost model $\mathcal{M}_{\text{XGB}}$ on $\mathcal{F}_{\text{best}}$
Model Evaluation:
    Compute:
    $A_{\text{test}} = \frac{\text{TP+TN}}{\text{Total Samples}}$, ROC-AUC$_{\text{test}}$
Save Model:
    Save trained XGBoost model

the most relevant features.

1. Recursive Feature Elimination with Cross-Validation (RFECV) iteratively removes the least important features based on model performance, resulting in the feature set F_{RFECV}.

$$F_{\text{RFECV}} = \{f_1, f_2, \ldots, f_k\} \quad k \text{ is the number of selected feature}$$

2. LASSO (Least Absolute Shrinkage and Selection Operator) employs an L_1-norm regularization term to enforce sparsity, optimizing the loss function:

$$\min \|y - X\beta\|_2^2 + \lambda\|\beta\|_1,$$

Where λ controls the regularization strength,

$$F_{\text{LASSO}} = \{f_1, f_2, \ldots, f_m\},$$

Where $m \leq k$. yielding the feature set F_{LASSO}.

3. Principal Component Analysis (PCA) transforms the data into a new coordinate system, capturing 95% of the variance in the top components F_{PCA}.

$$F_{\text{PCA}} = \{PC_1, PC_2, \ldots, PC_p\},$$

4. Mutual Information (MI) measures dependency between features and the target variable, selecting features F_{MI} with the highest scores.

$$F_{\text{MI}} = \{f_1, f_2, \ldots, f_q\},$$

Where q is determined by a threshold τ.

The selected features from various feature selection methods F_{RFECV}, F_{LASSO}, F_{PCA}, and $F_{\text{MI}}$are fed into the XGBoost classifier for training. XGBoost (eXtreme Gradient Boosting) is a robust and efficient gradient-boosting framework known for its scalability and superior performance on structured datasets. In this IDS model, XGBoost was configured with an architecture comprising 100 boosting rounds ($n_{\text{estimators}} = 100$) to iteratively refine the model's accuracy. A learning rate ($\eta = 0.1$) was employed to ensure gradual learning during training, balancing convergence speed and stability.

The classifier's objective function includes regularization terms to prevent overfitting, defined as:

$$\mathcal{L}(\theta) = \sum_{k=1}^{n} \ell(y_k, \hat{y}_k) + \lambda_1 \|\theta\|_1 + \lambda_2 \|\theta\|_2^2, \tag{4.4}$$

Where $\ell(y_k, \hat{y}_k)$ is the loss function, and λ_1 and λ_2 are regularization parameters for L_1 (sparsity) and L_2 (generalization) penalties, respectively.

During training, the XGBoost model monitors log loss (*LogLoss*) and classification error. Log loss is defined as:

$$\text{LogLoss} = -\frac{1}{N} \sum_{k=1}^{N} \left[y_k \log(\hat{y}_k) + (1 - y_k) \log(1 - \hat{y}_k) \right], \tag{4.5}$$

Where N is the total number of samples, y_i is the i^{th} true label and $\hat{y}_i$ is the i^{th} predicted probability of the positive class. This metric provides real-time feedback to adjust the model during the boosting process.

With this setup, XGBoost leverages its ability to handle high-dimensional data and noisy environments effectively, making it an ideal choice for

detecting cyber threats in the AgriFedSecure framework. After completing the model training, performance metrics such as accuracy and ROC-AUC scores are computed to evaluate the system's effectiveness.

$$A_{\text{test}} = \frac{\text{TP} + \text{TN}}{\text{Total Samples}},$$

$$\text{ROC-AUC}_{\text{test}} = \int_0^1 \text{TPR}(\text{FPR})\, d\text{FPR},$$

Where TPR and FPR denote the true positive and false positive rates, respectively.

When combined with the Federated Learning (FL) model, the AgriFedSecure framework showcases a comprehensive solution that not only ensures accurate predictions for soil moisture levels but also maintains the security and privacy of the distributed training environment. This integration of FL and IDS strengthens the overall system's reliability, making it a highly effective tool for smart agriculture applications.

Chapter 5

Results and Discussions

This section presents a comprehensive evaluation of the proposed framework's overall performance using the Soil Moisture dataset[1] and the KDD Cup[2] 1999 datasets.

The Wazihub Soil Moisture Prediction Challenge dataset is an open-access dataset specifically created for time-series forecasting applications in smart agriculture. This dataset serves as the basis for training and testing predictive models. It includes time-stamped recordings of various soil and environmental attributes, which are used as input features for the local model. The features of the dataset are

- Soil Moisture 1, 2, 3, and 4: These represent the target variables indicating soil moisture content at different depths or locations.
- Air Temperature: Reflects the ambient temperature of the surrounding environment.
- Air Humidity: Indicates the relative humidity of the air.
- Pressure: Represents the atmospheric pressure at the time of measurement.
- Wind Speed: Captures the wind speed at the measurement location.
- Wind Gust: Refers to short and sudden bursts of increased wind speed.
- Wind Direction: Specifies the direction of wind flow.

[1] https://zindi.africa/competitions/wazihub-soil-moisture-prediction-challenge/data

[2] https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

5.1 Performance of local models

The results for each soil moisture category are presented with visualizations, along with key metrics such as R^2 coefficient of determination and Mean Squared Error (MSE). These metrics comprehensively demonstrate the predictive accuracy of the local Long Short-Term Memory (LSTM) models.

The trained local LSTM models exhibited outstanding predictive capabilities for soil moisture levels across all four categories. The diagrams illustrated in Figure 5.1, 5.2, 5.3, 5.4 are the comparison between predicted and actual soil moisture values across different datasets. In each graph, the actual values (orange line) are plotted against the predicted values (blue line), with closer alignment indicating higher prediction accuracy. The model successfully captures the overall trends in soil moisture variations, though some deviations are observed, particularly in regions with sudden fluctuations. While the LSTM model performs well in learning temporal dependencies, slight discrepancies appear in areas with sharp transitions, suggesting potential challenges in handling rapid changes. However, despite these small errors, the model is still a useful tool for predicting soil moisture, which can help farmers manage irrigation better and improve crop growth. Overall, the results show that the model works well and can be improved further to make even more accurate predictions.

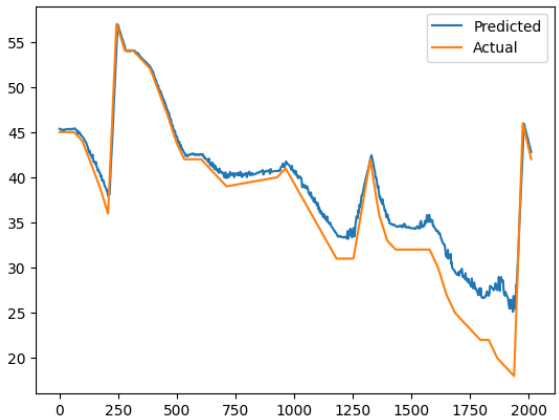

Figure 5.1: Predicted vs Actual of Soil Moisture 1

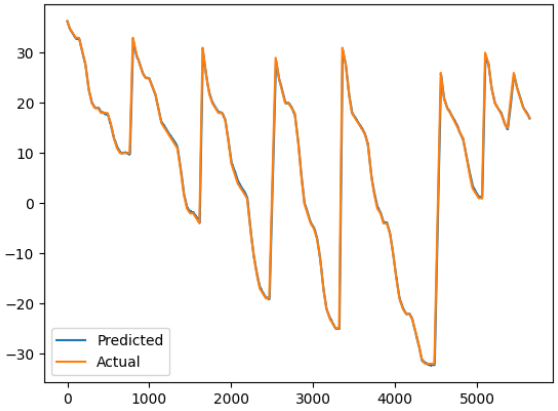

Figure 5.2: Predicted vs Actual of Soil Moisture 2

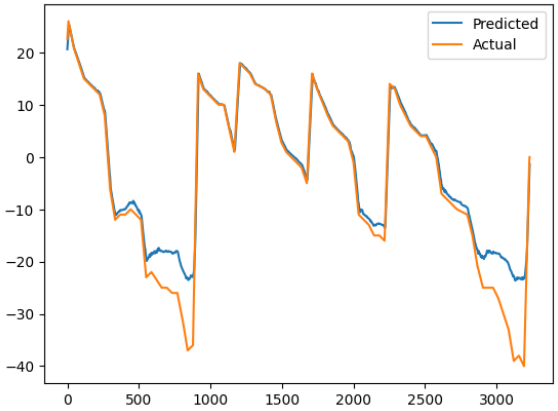

Figure 5.3: Predicted vs Actual of Soil Moisture 3

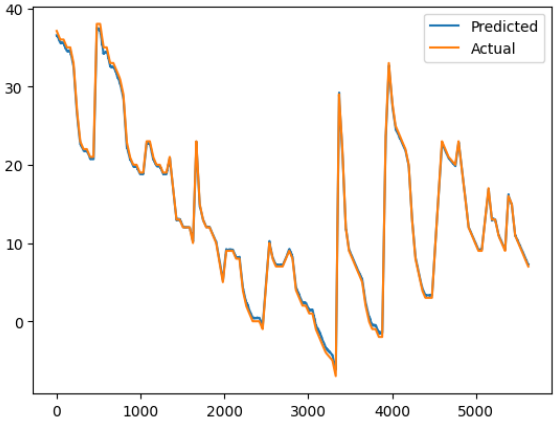

Figure 5.4: Predicted vs Actual of Soil Moisture 4

Table 5.2 shows that the performance Metrics of LSTM Models across all soil moisture datasets, achieving R^2 scores of 0.9847511649131775, 0.997300386428833, 0.9204849600791931, and 0.9990063905715942 with

Table 5.1: Performance Metrics of Local Models for Soil Moisture Prediction

Metric	**SM-1**	**SM-2**	**SM-3**	**SM-4**
R^2	0.9847	0.9973	0.9204	0.9990
MSE	1.1971	0.7811	20.7604	0.1111

corresponding MSE values of 1.1971534490585327, 0.7811405658721924, 20.760417938232422, and 0.11112716048955917 for soil moisture [SM] 1, 2, 3, and 4, respectively. These results underscore the model's efficiency in capturing temporal patterns and delivering precise predictions, making it a reliable tool for smart agriculture decision-making.

A previous implementation Google Colab User [2025] used XGBoost regression to predict soil moisture changes in agricultural fields. Although the approach was well-organized, it resulted in relatively low prediction accuracy, with R^2 values of 0.5158, 0.5328, 0.3165, and 0.4215. In contrast, my model, which uses an LSTM-based approach, achieved significantly better results. The R^2 values for my four local LSTM models were 0.9848, 0.9973, 0.9205, and 0.9990, respectively. These results show that the LSTM model was more effective at learning the patterns in the data and producing more accurate soil moisture predictions.

Table 5.2: Comparison Between Proposed Model R^2 vs Existing Model R^2

Metric	**SM-1**	**SM-2**	**SM-3**	**SM-4**
Proposed Model R^2	0.9847	0.9973	0.9204	0.9990
Existing Model R^2	0.5158	0.5328	0.3165	0.4215

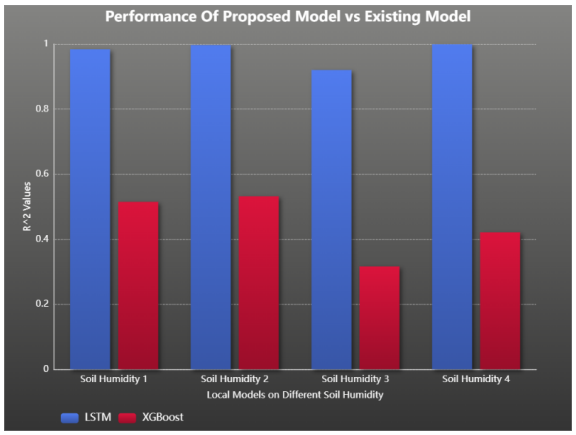

Figure 5.5: Comparison Between Proposed Model vs Existing Model

In the interest of time efficiency, only the Soil Moisture 2 and Soil Moisture 4 was selected for Federated Learning experiments, as it yielded the

best results. The dataset corresponding to Soil Moisture 2 was allocated to Client One, while the Soil Moisture 4 was assigned to Client Two. Both clients trained their respective local models using their dataset.

5.2 Performance of AgriFedSecure framework

The graphs illustrated in Figure 5.6, 5.7 show how well the Federated Learning model predicts soil moisture levels for two different clients. The blue line represents the predicted values, while the orange line shows the actual values. In the first graph, for Client One, the model achieved an R^2 score of 0.91, meaning the predictions are quite accurate, though there are some differences, especially when moisture levels change suddenly. In the second graph, for Client Two, the model performed even better, with an R^2 score of 0.95. The predicted values closely match the actual ones, but there are still some mismatches during sharp increases and drops. Overall, the model does a good job of capturing moisture trends, but it needs improvement to handle sudden changes more and more accurately.

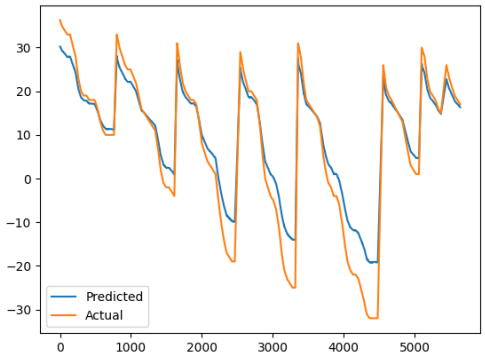

Figure 5.6: Client One Results

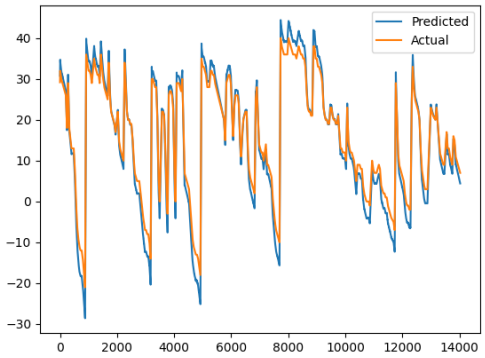

Figure 5.7: Client Two Results

Table 5.3: Federated Learning Results

Client	R^2	Description
Client One	0.91	Soil moisture 2 dataset
Client Two	0.95	Soil moisture 4 dataset

The results of the Federated Learning model for Soil Moisture 2 and 4 are summarized in Table 5.3. These results highlight the models ability to effectively predict soil moisture levels across distributed datasets while preserving data privacy. The predictions generated by this model provide actionable insights for agricultural experts, enabling them to make informed decisions, such as determining whether to water a plant or not. This capability underscores the potential of Federated Learning in supporting precision agriculture and optimizing resource utilization in a privacy-preserving manner.

An earlier approach Google Colab User [2025] which is a non Fl based approach used XGBoost regression to predict soil moisture, but it showed low accuracy with R^2 values of 0.5328 for Soil moisture 2 and 0.4215 for Soil moisture 4. In comparison, my federated learning (FL) model using LSTM performed much better. Client One, trained on the Soil moisture 2 dataset, achieved an R^2 of 0.91, while Client Two, using Soil moisture 4, reached 0.95. This shows that the FL-based LSTM model provided more accurate predictions than a non Fl based approach.

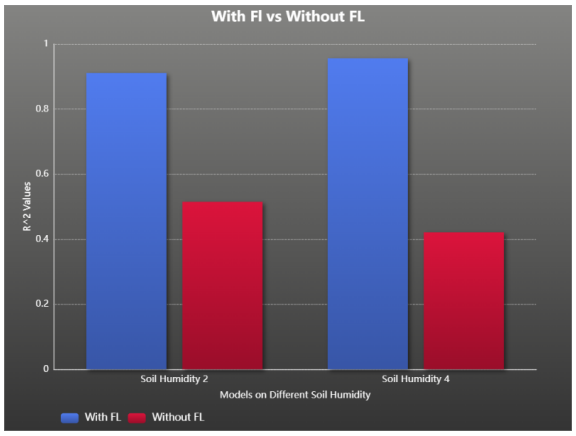

Figure 5.8: Comparison Between Fl Based Approach vs Non Fl Based Approach

5.3 Performance of intrusion detection system

The development of the IDS model is a systematic and carefully structured process designed to enhance the detection of potential cyber threats. This IDS model is trained and tested on the KDD Cup 1999 dataset, a benchmark dataset widely used for evaluating intrusion detection systems. The approach involves a sequence of well-defined steps that ensure efficient data preprocessing, optimal feature selection, and accurate evaluation of the models performance.

Comparison of Feature Selection Methods:

	Method	Accuracy	ROC-AUC
0	RFECV	0.95647	0.990867
1	LASSO	0.955941	0.98943
2	PCA	0.954618	0.986886
3	Mutual Information	0.948134	0.98685

Figure 5.9: Comparison of four different Feature selection methods

```
--- XGBoost Model ---
Accuracy: 0.9959
ROC-AUC Score: 0.9999
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      1.00      3516
           1       1.00      1.00      1.00      4042

    accuracy                           1.00      7558
   macro avg       1.00      1.00      1.00      7558
weighted avg       1.00      1.00      1.00      7558
```

Figure 5.10: Evaluation of Intrusion Detection System using XGBoost

The Intrusion Detection System (IDS) component of the AgriFedSecure framework was evaluated using multiple feature selection methods: RFECV, LASSO, PCA, and Mutual Information. These methods were applied to preprocess the data and identify the most relevant features for training robust models. The selected features were subsequently used to train classification models, and their performance was assessed using accuracy and the ROC-AUC score. The results from all four feature selection methods were summarized in Figure 5.9. RFECV emerged as the most effective method, achieving the highest accuracy and ROC-AUC scores.

The best feature set identified through feature selection was used to train an XGBoost classifier. The model was trained with 100 boosting rounds, a learning rate of 0.1, and log loss as the optimization objective to mitigate overfitting. The XGBoost classifier achieved 99.5% accuracy and 99.9% ROC-AUC score as shown in the below Figure 5.10

The above classification report provides key details for each class. The high precision and recall values indicate that the model makes very few mistakes in classifying data.

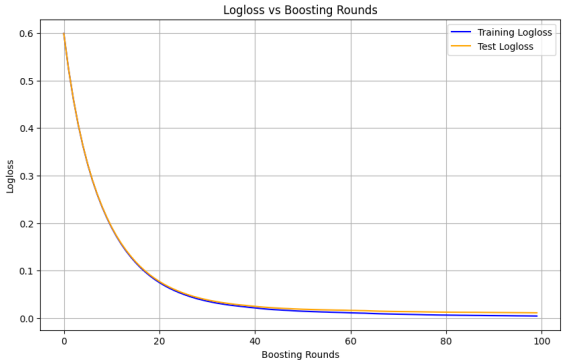

Figure 5.11: Performance of Model's Prediction Error

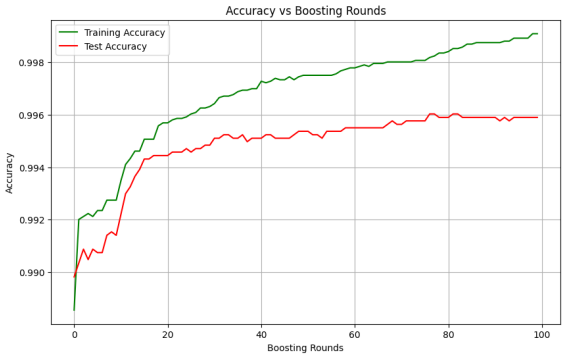

Figure 5.12: Performance of Models Classification

The above figures illustrate the training progress and performance of the XGBoost model over multiple boosting rounds. Figure 5.11 which plots log loss vs. boosting rounds, shows how the model's prediction errors decrease as training progresses. Log loss is a measure of how well the predicted probabilities match the actual labels, with lower values indicating better performance. The blue line represents the training log loss, while the orange line represents the test log loss. Initially, both values are high, but they steadily decrease as the number of boosting rounds increases. Around 40 boosting rounds, the log loss becomes very small and stabilizes, suggesting that the model has effectively learned the patterns in the data and is not overfitting.

The Figure 5.12 presents accuracy vs. boosting rounds, showing how the models classification performance improves over time. The green line represents training accuracy, while the red line represents test accuracy. Both start at around 99.0% and gradually increase as boosting progresses. After approximately 40 rounds, test accuracy stabilizes at 99.4%, while training accuracy continues to improve slightly, reaching about 99.8%. The small difference between training and test accuracy shows that the model performs well on new data without much overfitting.

The results indicate that integrating feature selection techniques with advanced classifiers like XGBoost provides highly accurate anomaly detection capabilities. The models ability to detect malicious activities with near-perfect metrics ensures robust security for the AgriFedSecure Framework.

Chapter 6

Conclusion

6.1 Conclusion and prospects for future Work

In this research, we presented and designed the AgriFedSecure framework, which combines federated learning (FL), and intrusion detection systems (IDS) to address the challenges of soil moisture prediction and anomaly detection. The local models achieved an impressive R^2 score of [0.98,0.99,0.92,0.99], while the FL model further enhanced performance with R^2 scores of [0.91,0.95]. To ensure the security and integrity of the FL model, the framework incorporates IDS at both the client and server sides. This robust integration ensures the detection of anomalies with 99% accuracy, effectively countering above mentioned potential attacks. Looking ahead, future work could focus on enhancing the AgriFedSecure framework by expanding IDS capabilities to handle a wider range of sophisticated threats. Furthermore, improvements in the FL model could be achieved by adopting advanced aggregation methods such as qFedAvg (Quantile Federated Averaging) and Per-FedAvg (Personalized Federated Averaging), which are designed to optimize performance for heterogeneous datasets. Additionally, integrating other robust and cutting-edge technologies for mitigating cyber threats can further strengthen the framework, making it even more resilient and effective for smart agriculture applications.

Bibliography

[1] Usama Ahmed, Mohammad Nazir, Amna Sarwar, Tariq Ali, El-Hadi M Aggoune, Tariq Shahzad, and Muhammad Adnan Khan. Signature-based intrusion detection using machine learning and deep learning approaches empowered with fuzzy clustering. *Scientific Reports*, 15(1):1726, 2025.

[2] Tamara Al-Shurbaji, Mohammed Anbar, Selvakumar Manickam, Iznan H Hasbullah, Nadia ALfriehate, Basim Ahmad Alabsi, Ahmad Reda Alzighaibi, and Hasan Hashim. Deep learning-based intrusion detection system for detecting iot botnet attacks: A review. *IEEE Access*, 2025.

[3] Nemalikanti Anand, Saifulla MA, Pavan Kumar Aakula, Raveendra Babu Ponnuru, Rizwan Patan, and Chegireddy Rama Prakasha Reddy. Enhancing intrusion detection against denial of service and distributed denial of service attacks: Leveraging extended berkeley packet filter and machine learning algorithms. *IET Communications*, 19(1):e12879, 2025.

[4] Mohammad Arafah, Iain Phillips, Asma Adnane, Wael Hadi, Mohammad Alauthman, and Abedal-Kareem Al-Banna. Anomaly-based network intrusion detection using denoising autoencoder and wasserstein gan synthetic attacks. *Applied Soft Computing*, 168:112455, 2025.

[5] Somnath Bera, Tanushree Dey, Anwesha Mukherjee, and Debashis De. Flag: Federated learning for sustainable irrigation in agriculture 5.0. *IEEE Transactions on Consumer Electronics*, 2024.

[6] Yu Cai, Wengang Zheng, Xin Zhang, Lili Zhangzhong, and Xuzhang Xue. Research on soil moisture prediction model based on deep learning. *PloS one*, 14 (4):e0214508, 2019.

[7] Vijaya Choudhary, Paramita Guha, Giovanni Pau, and Sunita Mishra. An overview of smart agriculture using internet of things (iot) and web services. *Environmental and Sustainability Indicators*, page 100607, 2025.

[8] Mohamed El Bouazzati, Philippe Tanguy, Guy Gogniat, and Russell Tessier. Diwall: A lightweight host intrusion detection system against jamming and packet injection attacks. *ACM Transactions on Embedded Computing Systems*, 2025.

[9] Google Colab User. Example google colab notebook. https://colab.research.google.com/drive/11L4OLJW-0TCUa3lYptvWi3Q3C4LNsJje?usp=sharing, 2025. Accessed: 2025-05-31.

[10] D Hemkumar. Preserving location privacy against inference attacks in indoor positioning system. *Peer-to-Peer Networking and Applications*, 17(2):784–799, 2024.

[11] D Hemkumar, S Ravichandra, and DVLN Somayajulu. Impact of prior knowledge on privacy leakage in trajectory data publishing. *Engineering Science and Technology, an International Journal*, 23(6):1291–1300, 2020.

[12] D Hemkumar, Sadam Ravichandra, and Durvasula VLN Somayajulu. Impact of data correlation on privacy budget allocation in continuous publication of location statistics. *Peer-to-Peer Networking and Applications*, 14:1650–1665, 2021.

[13] Prabhat Kumar, Govind P Gupta, and Rakesh Tripathi. Pefl: Deep privacy-encoding-based federated learning framework for smart agriculture. *IEEE Micro*, 42(1):33–40, 2021.

[14] Vivek Kumar, Kamal Kumar, Maheep Singh, and Neeraj Kumar. Nids-da: Detecting functionally preserved adversarial examples for network intrusion detection system using deep autoencoders. *Expert Systems with Applications*, page 126513, 2025.

[15] N Sharmila Kumari, HS Vimala, CN Pruthvi, and J Shreyas. Securing iot: Designing a resilient probing route against sinkhole attacks. In *Data Science & Exploration in Artificial Intelligence: Proceedings of the First International Conference On Data Science & Exploration in Artificial Intelligence (CODE-AI 2024) Bangalore, India, 3rd-4th July, 2024 (Volume 2)*, page 351. CRC Press, 2025.

[16] Sahar Lazim and Qutaiba I Ali. Machine learning-based intrusion detection and prevention system for iiot smart metering networks: Challenges and solutions. *arXiv preprint arXiv:2502.11138*, 2025.

[17] Kondru Mounika, P Venkateswara Rao, and Anand Anbalagan. Modified cnn model for network intrusion detection and classification system using local outlier factor-based recursive feature elimination.

[18] Christopher Olah. Understanding lstm networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015. Accessed: 2025-05-27.

[19] Mrutyunjay Padhiary and Raushan Kumar. Enhancing agriculture through ai vision and machine learning: The evolution of smart farming. In *Advancements in Intelligent Process Automation*, pages 295–324. IGI Global, 2025.

[20] Mahfuzur Rahman, Md Mehedi Hasan, Md Anuwer Hossain, Utpal Kanti Das, Md Monirul Islam, Mohammad Rezaul Karim, Hamid Faiz, Zulfiqar Hammad, Shamsher Sadiq, and Mehtab Alam. Integrating deep learning algorithms for forecasting evapotranspiration and assessing crop water stress in agricultural water management. *Journal of Environmental Management*, 375:124363, 2025.

[21] M Rajkumar, J Karthika, et al. Multi-view consistent generative adversarial network for enhancing intrusion detection with prevention systems in mobile

ad hoc networks against security attacks. *Computers & Security*, 150:104242, 2025.

[22] Fitsum T Teshome, Haimanote K Bayabil, Bruce Schaffer, Yiannis Ampatzidis, and Gerrit Hoogenboom. Improving soil moisture prediction with deep learning and machine learning models. *Computers and Electronics in Agriculture*, 226: 109414, 2024.

[23] Duy Thanh Tran, Hoanh Su Le, and Jun-Ho Huh. Building an automatic irrigation fertilization system for smart farm in greenhouse. *IEEE Transactions on Consumer Electronics*, 2023.

[24] Waziup. Wazihub_competition.ipynb. GitHub repository: ZindiAICompetition, 2025. URL https://github.com/Waziup/ZindiAICompetition/blob/master/winners/WinnersSoln/Wazihub_competition.ipynb. Accessed: 2025-05-23.

[25] Qiang Yang, Yang Liu, Tianjian Chen, and Yong Cheng. Federated learning. https://www.scribd.com/document/863784236/Federated-Learning-Qiang-Yang-Yang-Liu-Tianjian-Chen-Yong-Cheng-download, 2021. Available at https://www.scribd.com/document/863784236/Federated-Learning-Qiang-Yang-Yang-Liu-Tianjian-Chen-Yong-Cheng-download.

[26] Limin Yu, Wanlin Gao, Redmond R Shamshiri, Sha Tao, Yanzhao Ren, Yanjun Zhang, and Guilian Su. Review of research progress on soil moisture sensor technology. *International Journal of Agricultural and Biological Engineering*, 14(4):32–42, 2021.

Research Paper Submission

As part of this project work, I and Dr.Hemkumar D authored a research paper titled **"Accurate Soil Humidity Prediction Using Federated Learning"**, which was **accepted** for full paper presentation (Paper ID: 311) at **The Seventh International Conference on Computational Intelligence in Communications and Business Analytics (CICBA 2025)**, held at Jadavpur University, Kolkata, India, from 04th to 06th July, 2025. The presentation is scheduled on July 5th, 2025.